UNIVERSIDADE ESTADUAL DE PONTA GROSSA
SETOR DE ENGENHARIAS, CIÊNCIAS AGRÁRIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

GABRIEL DO ESPÍRITO SANTO
PEDRO AUGUSTO MARQUES KLOSTER

PROJETO DE BANCO DE DADOS

PONTA GROSSA
2025

GABRIEL DO ESPÍRITO SANTO
PEDRO AUGUSTO MARQUES KLOSTER

PROJETO DE BANCO DE DADOS

Trabalho apresentado à professora Maria Salete Marcon Gomes Vaz para obtenção de nota na disciplina de Banco de Dados no curso de Engenharia de Computação.

PONTA GROSSA
2025

RESUMO

Este trabalho apresenta o desenvolvimento completo de um sistema de banco de dados para a gestão integrada de uma rede de academias. A elaboração iniciou-se com a modelagem conceitual por meio do Diagrama Entidade–Relacionamento (E/R), seguido pelo mapeamento para o modelo relacional e pela implementação física utilizando SQL. O sistema contempla os principais processos administrativos da academia, como cadastro de alunos, controle biométrico de acesso via catraca, gerenciamento de planos, registro de pagamentos, integração com o servidor externo Gympass e armazenamento do histórico de avaliações físicas realizadas por professores.

A implementação inclui a criação de tabelas, chaves, domínios e diversas restrições de integridade, garantindo consistência e segurança dos dados. Além disso, foram desenvolvidas views, triggers e stored procedures que automatizam regras de negócio, padronizam operações e auxiliam na geração de relatórios e monitoramento das atividades internas. O dicionário de dados documenta todos os elementos estruturais do banco, oferecendo clareza para manutenção futura.

ABSTRACT

This work presents the complete development of a database system for the integrated management of a network of gyms. The development began with conceptual modeling using the Entity-Relationship (E/R) Diagram, followed by mapping to the relational model and physical implementation using SQL. The system covers the main administrative processes of the gym, such as student registration, biometric access control via turnstile, plan management, payment registration, integration with the external Gympass server, and storage of the history of physical assessments performed by instructors.

The implementation includes the creation of tables, keys, domains, and various integrity constraints, ensuring data consistency and security. In addition, views, triggers, and stored procedures were developed to automate business rules, standardize operations, and assist in generating reports and monitoring internal activities. The data dictionary documents all the structural elements of the database, providing clarity for future maintenance.

SUMÁRIO

1. Introdução

A elaboração de um banco de dados bem estruturado é essencial para garantir a organização, integridade e disponibilidade das informações em sistemas computacionais. Este trabalho apresenta o desenvolvimento completo de um sistema de gerenciamento para uma rede de academias, abordando todas as fases do processo (desde a análise dos requisitos até a implementação prática em SQL), com foco na construção de uma solução coerente e funcional.

A aplicação proposta envolve diferentes elementos do ambiente da academia, como alunos, recepcionistas, professores, catracas com identificação biométrica, máquinas de pagamento e integração com serviços externos, como o Gympass. Esses componentes definem os fluxos de operação do sistema, como cadastros, validações de acesso, avaliações físicas, registros de pagamento e controle de planos, permitindo identificar os dados que precisam ser armazenados e as regras que orientam o comportamento da aplicação.

Com base nesses requisitos, foi elaborado o Esquema Entidade–Relacionamento (E/R). A partir do E/R, foi desenvolvido o Esquema Relacional. A etapa seguinte compreende a implementação em SQL, na qual o banco de dados é de fato construído utilizando o SGBD PostrgreSQL. Inclui-se a criação de tablespaces, criação de tabelas, e definição de restrições de integridade. Além disso, foram desenvolvidas views, triggers, e stored procedures.

Por fim, o trabalho contempla a criação de diversas consultas SQL, elaboradas para atender às necessidades da academia. Essas consultas permitem gerar relatórios, integrar dados entre alunos internos e externos, verificar acessos, acompanhar históricos de avaliações físicas e monitorar pagamentos, planos e registros feitos pelos recepcionistas.

2. Embasamento teórico

2.1 Banco de dados

Um sistema de gerenciamento de banco de dados (SGBD) é a camada de software responsável por armazenar, recuperar e garantir a consistência das informações de uma aplicação. Os bancos de dados relacionais organizam dados em tabelas e fornecem mecanismos para manipulação (SQL), consistência e controle de concorrência. (Oracle, 2021)

2.2 Modelagem conceitual: Diagrama Entidade–Relacionamento (E/R)

A modelagem conceitual é a primeira fase do projeto de banco de dados e tem por objetivo representar o domínio do problema independentemente de detalhes de implementação. O Modelo Entidade–Relacionamento (E/R) é uma notação consagrada para descrever entidades, atributos e relacionamentos, incluindo cardinalidades e restrições semânticas. Ele facilita a comunicação e serve de base para o mapeamento ao modelo relacional. Ao documentar a academia (alunos, recepcionistas, professores, avaliações, catracas, servidor externo Gympass etc.), o E/R permite capturar regras como “cada aluno tem um plano ativo por vez” ou “avaliações são registradas por professor e recepcionista”. (Watt, 2014)

2.3. Do modelo conceitual para o relacional (mapeamento)

O mapeamento transforma entidades em tabelas, atributos em colunas e

relacionamentos em chaves estrangeiras (ou tabelas associativas para relacionamentos N:N). É nessa etapa que se definem chaves primárias, cardinalidades e integridade referencial: por exemplo, a entidade Aluno vira a tabela Aluno(cod_a, nome, cpf, ...) e um relacionamento “Aluno - Plano” é representado por cod_plano como chave estrangeira em Aluno (ou por tabela intermediária, dependendo do caso). O mapeamento também considera restrições de unicidade, obrigatoriedade e regras de exclusão/atualização, que serão implementadas em linguagem SQL. (Song et al., 1995)

2.4 Integridade semântica e restrições (checks, domínios, chaves)

Além de integridade referencial e chaves primárias, existem restrições declarativas que restringem o domínio de valores: CHECK (condições booleanas), NOT NULL, UNIQUE, e tipos/domínios customizados. Essas restrições previnem inserções inválidas (ex.: preço do plano ≥ 0, datas coerentes). (Doorn and Rivero, 2001)

2.5 Mecanismos procedurais: Triggers e Stored Procedures

Triggers são rotinas armazenadas que o SGBD executa automaticamente em resposta a eventos DML (INSERT, UPDATE, DELETE). Elas são usadas para auditoria (registrar quem e quando alterou), manter derivadas (atualizar campos calculados), e impor regras complexas não facilmente expressas por constraints. Stored procedures são sub-rotinas armazenadas que encapsulam lógica de aplicação no banco (validações complexas, processos de cobrança, integrações), permitindo reutilização, segurança (controlar acesso via execução) e otimização de rede (execução no servidor). (Cochrane, R., 2000)

2.6 Views

Views são tabelas virtuais definidas por consultas SQL armazenadas no SGBD. Elas oferecem abstração e segurança (ex.: expor apenas colunas necessárias a determinados perfis), simplificam consultas recorrentes e permitem criar relatórios consolidados sem duplicar dados. Algumas views são “materializadas” (armazenam resultados) para melhorar performance em consultas pesadas, ao custo de manutenção extra. A modelagem de views deve considerar atualizabilidade e custos de manutenção. (Chen, 2004)

2.7 Implementação física: tablespaces, arquivos e organização de dados

Tablespaces são unidades lógicas de armazenamento que agrupam objetos de banco (tabelas, índices) e mapeiam para arquivos físicos no sistema operacional. Eles permitem separar tipos de dados (índices vs. dados) e administrar espaço, backups e desempenho. O planejamento de tablespaces auxilia em escalabilidade e manutenção (ex.: colocar índices em tablespaces separados para otimizar I/O). Documentação técnica das implementações detalha práticas de gerenciamento de tablespaces. (Oracle Brasil, 2019)

2.8 Transações, concorrência e propriedades ACID

Transações agrupam operações de forma a serem atômicas: todas ocorrem ou nenhuma. Para aplicações multiusuário é fundamental garantir isolamento entre transações concorrentes (evitar leituras sujas, não repetíveis e leituras fantasma) e durabilidade (garantir que commits sobrevivam a falhas). As propriedades ACID são o pilar para sistemas transacionais (pagamentos, atualizações de planos, registros

de acesso), e o SGBD fornece mecanismos (locks, logs de recuperação) para implementá-las. (Pykes, 2025)

2.9 Índices e desempenho de consultas

Índices (B-tree, hashing, índices compostos) aceleram buscas e junções, mas ocupam espaço e encarecem operações de escrita. O projeto de índices deve seguir o padrão de consultas (WHERE, JOIN, ORDER BY) observadas nas consultas esperadas (relatórios de alunos, histórico de avaliações, verificação de status do Gympass). Estratégias como índices parciais, índices sobre colunas seletivas e análise de plano de execução ajudam a balancear desempenho. Ferramentas de monitoramento do SGBD permitem identificar gargalos e ajustar índices. (fonte geral de práticas de tuning de RDBMS). (Trillo-Montero, D., et al., 2023)

2.10 Integração com sistemas externos (Gympass) e aspectos de segurança

Integrações com servidores externos (por exemplo, validação de Gympass) exigem modelagem de dados para representar tokens/códigos de acesso, logs de validação e tratamento de erros. É importante registrar eventos de integração para auditoria e reconciliar estados (ex.: um código Gympass validado externamente, mas com falha de gravação local). (ScienceDirect Topics, 2016)

2.11 Dicionário de dados e documentação

O dicionário de dados reúne metadados: definição de tabelas, colunas, tipos, restrições, relacionamentos, índices, views e procedures. Uma boa documentação facilita manutenção, onboarding e auditorias. Além disso, manter um histórico de alterações (versão de esquema) é prática essencial em projetos acadêmicos e profissionais.

3. Desenvolvimento de Aplicação de Banco de Dados

3.1. Descrição da Aplicação

Uma rede de academias deseja informatizar seu processo de gestão por meio de um sistema de gerenciamento acadêmico. Esse sistema envolve diferentes integrantes: alunos, professores, recepcionistas e um servidor externo responsável pela integração com o Gympass.

O processo de cadastro de alunos é realizado exclusivamente pela recepcionista. Para cada aluno, devem ser armazenados os seguintes dados:

- Nome completo
- CPF
- Data de nascimento
- Registro da digital

- Plano contratado (cada aluno pode estar vinculado a apenas um plano ativo por vez)

O acesso dos alunos à academia é feito por meio de catracas, que possuem um leitor biométrico. Esse dispositivo identifica a digital cadastrada e verifica se o plano associado ao aluno está ativo (“em dia”). Caso esteja regular, a entrada é liberada.

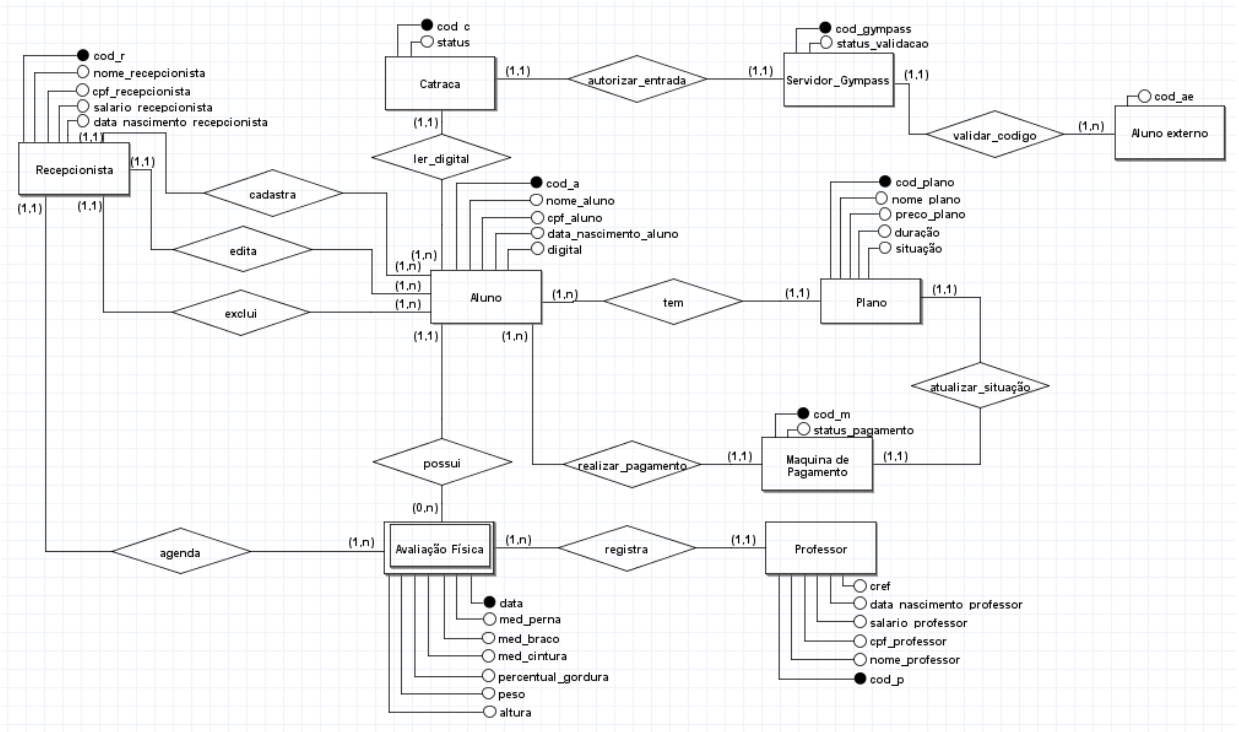
Além disso, alunos que não se cadastraram diretamente na academia podem acessar as instalações utilizando o Gympass. Nessa modalidade, o servidor externo gera um código único de acesso para cada visita, que é validado pelo sistema da academia no momento da entrada.

Para acompanhamento da evolução do aluno, o sistema deve registrar avaliações físicas periódicas. O agendamento dessas avaliações é feito pela recepcionista, enquanto o professor é responsável por lançar os resultados no sistema. Os atributos registrados em cada avaliação são: altura, peso, percentual de gordura, medida da cintura, dos braços e das pernas.

O pagamento dos planos é realizado presencialmente, utilizando cartão de crédito na máquina de pagamento da academia. Quando o sistema confirma a transação, o estado do plano do aluno passa automaticamente para “em dia”.

Por fim, o sistema deve permitir à recepcionista editar ou excluir cadastros de alunos sempre que necessário.

3.2. Esquema Entidade/Relacionamento



3.3. Esquema Relacional (Mapeamento)

Recepcionista(***cod_r,*** nome_recepcionista, cpf_recepcionista, salario_recepcionista, data_nascimento_recepcionista)

Catraca(***cod_c***, status, ***cod_gympass***)

Aluno(***cod_a***, nome_aluno, cpf_aluno, data_nascimento_aluno, digital, ***cod_r, cod_c, cod_m, cod_plano***)

Aluno_externo(***cod_ae, cod_gympass***)

Servidor_Gympass(***cod_gympass***, status_validacao)

Maquina_de_Pagamento(***cod_m***, status_pagamento, ***cod_plano***)

Professor(***cod_p***, cref, data_nascimento_professor, salario_professor, cpf_professor, nome_professor)

Avaliação_Física(***data, cod_a***, med_perna, med_braço, med_cintura, percentual_gordura, peso, altura, ***cod_r***, ***cod_p***)

Plano(***cod_plano,*** nome_plano, preco_plano, duração, situação)

3.4 Implementação SQL

3.4.1 Script SQL que deleta as tabelas modeladas caso já existam:

```sql
/* DELETANDO AS TABELAS CASO JÁ EXISTAM */

DROP TABLE Avaliacao_Fisica CASCADE;
DROP TABLE Aluno CASCADE;
DROP TABLE Recepcionista CASCADE ;
DROP TABLE Catraca  CASCADE;
DROP TABLE Aluno_externo CASCADE;
DROP TABLE Servidor_Gympass CASCADE ;
DROP TABLE Plano CASCADE;
DROP TABLE Maquina_de_Pagamento CASCADE;
DROP TABLE Professor CASCADE;
```

3.4.2 Script SQL que cria as tabelas modeladas no diagrama relacional:

```sql
/* CRIANDO AS TABELAS */

-- Tabela de Recepcionista

CREATE TABLE Recepcionista (
    cod_r INT PRIMARY KEY,
    nome_recepcionista VARCHAR(100) NOT NULL,
    cpf_recepcionista CHAR(11) UNIQUE NOT NULL,
    salario_recepcionista DECIMAL(10,2),
    data_nascimento_recepcionista DATE
);

-- Tabela de Servidor Gympass

CREATE TABLE Servidor_Gympass (
    cod_gympass INT PRIMARY KEY,
    status_validacao VARCHAR(20) NOT NULL
);

-- Tabela de Catraca

CREATE TABLE Catraca (
    cod_c INT PRIMARY KEY,
    status VARCHAR(20) NOT NULL,
    cod_gympass INT,
    FOREIGN KEY (cod_gympass) REFERENCES Servidor_Gympass(cod_gympass)
);

-- Tabela de Plano

CREATE TABLE Plano (
    cod_plano INT PRIMARY KEY,
    nome_plano VARCHAR(50) NOT NULL,
    preco_plano DECIMAL(10,2) NOT NULL,
    duracao INT NOT NULL,
    situacao VARCHAR(20) NOT NULL
);

-- Tabela de Máquina de Pagamento
```

```sql
CREATE TABLE Maquina_de_Pagamento (
    cod_m INT PRIMARY KEY,
    status_pagamento VARCHAR(20) NOT NULL,
    cod_plano INT NOT NULL,
    FOREIGN KEY (cod_plano) REFERENCES Plano(cod_plano)
);
-- Tabela de Aluno

CREATE TABLE Aluno (
    cod_a INT PRIMARY KEY,
    nome_aluno VARCHAR(100) NOT NULL,
    cpf_aluno CHAR(11) UNIQUE NOT NULL,
    data_nascimento_aluno DATE NOT NULL,
    digital VARCHAR(200),
    cod_r INT NOT NULL,
    cod_c INT NOT NULL,
    cod_m INT NOT NULL,
    cod_plano INT NOT NULL,
    FOREIGN KEY (cod_r) REFERENCES Recepcionista(cod_r),
    FOREIGN KEY (cod_c) REFERENCES Catraca(cod_c),
    FOREIGN KEY (cod_m) REFERENCES Maquina_de_Pagamento(cod_m),
    FOREIGN KEY (cod_plano) REFERENCES Plano(cod_plano)
);

-- Tabela de Aluno Externo (Gympass)

CREATE TABLE Aluno_externo (
    cod_ae INT PRIMARY KEY,
    cod_gympass INT NOT NULL,
    FOREIGN KEY (cod_gympass) REFERENCES Servidor_Gympass(cod_gympass)
);

-- Tabela de Professor

CREATE TABLE Professor (
    cod_p INT PRIMARY KEY,
    cref VARCHAR(20) UNIQUE NOT NULL,
    data_nascimento_professor DATE NOT NULL,
    salario_professor DECIMAL(10,2),
    cpf_professor CHAR(11) UNIQUE NOT NULL,
    nome_professor VARCHAR(100) NOT NULL
);
```

```sql
-- Tabela de Avaliação Física

CREATE TABLE Avaliacao_Fisica (
    data DATE NOT NULL,
    cod_a INT NOT NULL,
    med_perna DECIMAL(5,2),
    med_braco DECIMAL(5,2),
    med_cintura DECIMAL(5,2),
    percentual_gordura DECIMAL(5,2),
    peso DECIMAL(5,2),
    altura DECIMAL(5,2),
    cod_r INT NOT NULL,
    cod_p INT NOT NULL,
    PRIMARY KEY (data, cod_a),
    FOREIGN KEY (cod_a) REFERENCES Aluno(cod_a),
    FOREIGN KEY (cod_r) REFERENCES Recepcionista(cod_r),
    FOREIGN KEY (cod_p) REFERENCES Professor(cod_p)
);
```

3.4.3 Restrições de integridade

A seguir, são criadas as restrições de integridade para o sistema:

```sql
/* CRIANDO AS RESTRIÇÕES DE INTEGRIDADE */

-- Recepcionista

ALTER TABLE Recepcionista -- Garante que o salário não seja negativo
    ADD CONSTRAINT ck_rec_salario CHECK (salario_recepcionista >= 0);

ALTER TABLE Recepcionista -- Garante uma data de nascimento "lógica"
(ninguém nascido antes de 1900)
    ADD CONSTRAINT ck_rec_nascimento CHECK
(data_nascimento_recepcionista > '1900-01-01');

-- Professor

ALTER TABLE Professor -- Garante uma data de nascimento "lógica"
(ninguém nascido antes de 1900)
```

```sql
    ADD CONSTRAINT ck_prof_nascimento CHECK (data_nascimento_professor
> '1900-01-01');

ALTER TABLE Professor -- Garante que o salário do professor não seja
negativo.
    ADD CONSTRAINT ck_prof_salario CHECK (salario_professor >= 0);

ALTER TABLE Professor -- Garante que o CREF tenha um tamanho mínimo
(evita erros de digitação com 1 ou 2 caracteres).
    ADD CONSTRAINT ck_prof_cref_len CHECK (LENGTH(cref) >= 4);

ALTER TABLE Professor -- Restrição do teto do  salário
    ADD CONSTRAINT ck_prof_salario_teto CHECK (salario_professor <
50000);

-- Plano

ALTER TABLE Plano  -- O preço do plano não pode ser negativo.
    ADD CONSTRAINT ck_plano_preco CHECK (preco_plano >= 0);

ALTER TABLE Plano --A duração do plano (provavelmente em meses ou dias)
deve ser maior que zero.
    ADD CONSTRAINT ck_plano_duracao CHECK (duracao > 0);

ALTER TABLE Plano --  Limita a situação do plano a valores específicos
    ADD CONSTRAINT ck_plano_situacao CHECK (situacao IN ('Ativo',
'Inativo', 'Suspenso'));

ALTER TABLE Plano -- Não permite dois planos com o mesmo nome
    ADD CONSTRAINT uq_plano_nome UNIQUE (nome_plano);

-- Catraca

ALTER TABLE Catraca -- Define os estados possíveis da catraca, evitando
textos aleatórios.
    ADD CONSTRAINT ck_catraca_status CHECK (status IN ('Liberado',
'Bloqueado', 'Manutencao'));

-- Servidor Gympass

ALTER TABLE Servidor_Gympass --  Padroniza os status de validação
retornados pelo servidor.
```

```sql
    ADD CONSTRAINT ck_gym_status CHECK (status_validacao IN ('Valido',
'Invalido', 'Erro'));

--  Maquina pagamento

ALTER TABLE Maquina_de_Pagamento -- Padroniza os status de pagamento
possíveis.
    ADD CONSTRAINT ck_maq_status CHECK (status_pagamento IN
('Aprovado', 'Recusado', 'Pendente'));

-- Aluno

ALTER TABLE Aluno -- Garante que a data de nascimento do aluno seja
válida.
    ADD CONSTRAINT ck_aluno_nasc CHECK (data_nascimento_aluno >
'1900-01-01');

ALTER TABLE Aluno -- Se a digital for preenchida, garante que não seja
uma string vazia ('').
    ADD CONSTRAINT ck_aluno_digital CHECK (LENGTH(digital) > 0);

-- Avaliacao fisica

ALTER TABLE Avaliacao_Fisica -- O percentual de gordura deve estar
logicamente entre 0 e 100.
    ADD CONSTRAINT ck_av_gordura CHECK (percentual_gordura >= 0 AND
percentual_gordura <= 100);

ALTER TABLE Avaliacao_Fisica  -- O peso do aluno deve ser positivo.
    ADD CONSTRAINT ck_av_peso CHECK (peso > 0);

ALTER TABLE Avaliacao_Fisica -- A altura deve ser positiva e
logicamente aceitável (ex: menor que 3 metros).
    ADD CONSTRAINT ck_av_altura CHECK (altura > 0 AND altura < 3.00);

ALTER TABLE Avaliacao_Fisica -- Medida da perna deve ser positiva.
    ADD CONSTRAINT ck_av_perna CHECK (med_perna > 0);

ALTER TABLE Avaliacao_Fisica -- Mesma coisa para o braço
    ADD CONSTRAINT ck_av_braco CHECK (med_braco > 0);

ALTER TABLE Avaliacao_Fisica -- E cintura também
    ADD CONSTRAINT ck_av_cintura CHECK (med_cintura > 0);
```

```sql
ALTER TABLE Avaliacao_Fisica -- A cintura geralmente é menor que a
altura da pessoa. (Evita troca de campos na digitação).
    ADD CONSTRAINT ck_av_cintura_vs_braco CHECK (med_cintura >
med_braco);

ALTER TABLE Avaliacao_Fisica -- A data da avaliação física não pode ser
no futuro (ninguém faz avaliação amanhã).
    ADD CONSTRAINT ck_av_data_passado CHECK (data <= CURRENT_DATE);
```

3.4.4 - Script de inserção de dados

A seguir, está um código SQL para inserir dados e testar o sistema:

```sql
-- 1. Tabela RECEPCIONISTA
INSERT INTO Recepcionista (cod_r, nome_recepcionista,
cpf_recepcionista, salario_recepcionista,
data_nascimento_recepcionista) VALUES
(1, 'Ana Clara Souza', '11122233301', 2500.00, '1995-03-15'),
(2, 'Beatriz Lima', '22233344402', 2600.00, '1990-07-20'),
(3, 'Carlos Mendes', '33344455503', 2500.00, '1998-11-05'),
(4, 'Daniela Rocha', '44455566604', 2800.00, '1985-01-30'),
(5, 'Eduardo Silva', '55566677705', 2500.00, '2000-05-12');

-- 2. Tabela PROFESSOR
INSERT INTO Professor (cod_p, cref, data_nascimento_professor,
salario_professor, cpf_professor, nome_professor) VALUES
(1, '1234-G/PR', '1980-05-10', 4500.00, '99988877701', 'Roberto
Justus'),
(2, '5678-G/PR', '1992-08-22', 3800.00, '88877766602', 'Fernanda
Gentil'),
(3, '9012-G/PR', '1985-12-01', 4200.00, '77766655503', 'Paulo Muzy'),
(4, '3456-G/PR', '1995-02-14', 3500.00, '66655544404', 'Renato
Cariani'),
(5, '7890-G/PR', '1990-06-30', 4000.00, '55544433305', 'Juliana Paes');

-- 3. Tabela SERVIDOR_GYMPASS

INSERT INTO Servidor_Gympass (cod_gympass, status_validacao) VALUES
(100, 'Valido'), (101, 'Valido'), (102, 'Invalido'), (103, 'Valido'),
(104, 'Valido'),
(105, 'Erro'), (106, 'Valido'), (107, 'Invalido'), (108, 'Valido'),
(109, 'Valido'),
```

```
(110, 'Valido'), (111, 'Valido'), (112, 'Invalido'), (113, 'Valido'),
(114, 'Valido'),
(115, 'Erro'), (116, 'Valido'), (117, 'Valido'), (118, 'Invalido'),
(119, 'Valido');

-- 4. Tabela PLANO
INSERT INTO Plano (cod_plano, nome_plano, preco_plano, duracao,
situacao) VALUES
(1, 'Mensal Gold', 120.00, 30, 'Ativo'),
(2, 'Trimestral Power', 300.00, 90, 'Ativo'),
(3, 'Semestral Fit', 550.00, 180, 'Ativo'),
(4, 'Anual Pro', 1000.00, 365, 'Ativo');

-- 5. Tabela CATRACA
INSERT INTO Catraca (cod_c, status, cod_gympass) VALUES
(1, 'Liberado', NULL),      -- Catraca normal
(2, 'Liberado', NULL),      -- Catraca normal
(3, 'Liberado', 100);       -- Catraca integrada Gympass

-- 6. Tabela MAQUINA_DE_PAGAMENTO
INSERT INTO Maquina_de_Pagamento (cod_m, status_pagamento, cod_plano)
VALUES
(1, 'Aprovado', 1), (2, 'Aprovado', 2), (3, 'Aprovado', 4), (4,
'Aprovado', 1), (5, 'Pendente', 3),
(6, 'Aprovado', 2), (7, 'Recusado', 1), (8, 'Aprovado', 4), (9,
'Aprovado', 1), (10, 'Aprovado', 3),
(11, 'Aprovado', 1), (12, 'Aprovado', 2), (13, 'Aprovado', 4), (14,
'Aprovado', 1), (15, 'Pendente', 3),
(16, 'Aprovado', 2), (17, 'Recusado', 1), (18, 'Aprovado', 4), (19,
'Aprovado', 1), (20, 'Aprovado', 3),
(21, 'Aprovado', 1), (22, 'Aprovado', 2), (23, 'Aprovado', 4), (24,
'Aprovado', 1), (25, 'Pendente', 3),
(26, 'Aprovado', 2), (27, 'Recusado', 1), (28, 'Aprovado', 4), (29,
'Aprovado', 1), (30, 'Aprovado', 3);

-- 7. Tabela ALUNO
INSERT INTO Aluno (cod_a, nome_aluno, cpf_aluno, data_nascimento_aluno,
digital, cod_r, cod_c, cod_m, cod_plano) VALUES
(1, 'João da Silva', '10101010101', '1999-01-10', 'DIGITAL_HASH_1', 1,
1, 1, 1),
(2, 'Maria Oliveira', '20202020202', '1995-05-20', 'DIGITAL_HASH_2', 2,
2, 2, 2),
```

```
(3, 'Pedro Santos', '30303030303', '2000-12-15', 'DIGITAL_HASH_3', 1,
1, 3, 4),
(4, 'Lucas Ferreira', '40404040404', '1998-03-08', 'DIGITAL_HASH_4', 3,
2, 4, 1),
(5, 'Carla Dias', '50505050505', '1987-07-12', 'DIGITAL_HASH_5', 2, 1,
5, 3),
(6, 'Marcos Vinicius', '60606060606', '1992-09-30', 'DIGITAL_HASH_6',
4, 2, 6, 2),
(7, 'Julia Roberts', '70707070707', '1994-11-01', 'DIGITAL_HASH_7', 1,
1, 7, 1),
(8, 'Fernando Collor', '80808080808', '1980-02-25', 'DIGITAL_HASH_8',
5, 2, 8, 4),
(9, 'Gustavo Lima', '90909090909', '1996-06-18', 'DIGITAL_HASH_9', 3,
1, 9, 1),
(10, 'Andressa Suita', '11111111111', '1993-08-14', 'DIGITAL_HASH_10',
2, 2, 10, 3),
(11, 'Bruno Mars', '12121212121', '1990-01-01', 'DIGITAL_HASH_11', 1,
1, 11, 1),
(12, 'Lady Gaga', '13131313131', '1986-04-10', 'DIGITAL_HASH_12', 4, 2,
12, 2),
(13, 'Tom Cruise', '14141414141', '1975-07-03', 'DIGITAL_HASH_13', 5,
1, 13, 4),
(14, 'Will Smith', '15151515151', '1978-09-25', 'DIGITAL_HASH_14', 2,
2, 14, 1),
(15, 'Angelina Jolie', '16161616161', '1982-12-12', 'DIGITAL_HASH_15',
1, 1, 15, 3),
(16, 'Brad Pitt', '17171717171', '1970-10-30', 'DIGITAL_HASH_16', 3, 2,
16, 2),
(17, 'Leonardo DiCaprio', '18181818181', '1974-11-11',
'DIGITAL_HASH_17', 4, 1, 17, 1),
(18, 'Johnny Depp', '19191919191', '1963-06-09', 'DIGITAL_HASH_18', 5,
2, 18, 4),
(19, 'Robert Downey Jr', '21212121212', '1965-04-04',
'DIGITAL_HASH_19', 1, 1, 19, 1),
(20, 'Scarlett Johansson', '23232323232', '1984-11-22',
'DIGITAL_HASH_20', 2, 2, 20, 3),
(21, 'Chris Evans', '24242424242', '1981-06-13', 'DIGITAL_HASH_21', 3,
1, 21, 1),
(22, 'Chris Hemsworth', '25252525252', '1983-08-11', 'DIGITAL_HASH_22',
4, 2, 22, 2),
(23, 'Mark Ruffalo', '26262626262', '1967-11-22', 'DIGITAL_HASH_23', 5,
1, 23, 4),
```

```
(24, 'Jeremy Renner', '27272727272', '1971-01-07', 'DIGITAL_HASH_24',
1, 2, 24, 1),
(25, 'Elizabeth Olsen', '28282828282', '1989-02-16', 'DIGITAL_HASH_25',
2, 1, 25, 3),
(26, 'Paul Rudd', '29292929292', '1969-04-06', 'DIGITAL_HASH_26', 3, 2,
26, 2),
(27, 'Benedict Cumberbatch', '31313131313', '1976-07-19',
'DIGITAL_HASH_27', 4, 1, 27, 1),
(28, 'Tom Holland', '32323232323', '1996-06-01', 'DIGITAL_HASH_28', 5,
2, 28, 4),
(29, 'Chadwick Boseman', '34343434343', '1976-11-29',
'DIGITAL_HASH_29', 1, 1, 29, 1),
(30, 'Brie Larson', '35353535353', '1989-10-01', 'DIGITAL_HASH_30', 2,
2, 30, 3);

-- 8. Tabela ALUNO_EXTERNO
INSERT INTO Aluno_externo (cod_ae, cod_gympass) VALUES
(1, 101), (2, 103), (3, 104), (4, 106), (5, 108),
(6, 109), (7, 110), (8, 111), (9, 113), (10, 114);

-- 9. Tabela AVALIACAO_FISICA
INSERT INTO Avaliacao_Fisica (data, cod_a, med_perna, med_braco,
med_cintura, percentual_gordura, peso, altura, cod_r, cod_p) VALUES
('2024-01-10', 1, 55.0, 35.0, 80.0, 15.5, 75.0, 1.75, 1, 1),
('2024-02-15', 2, 50.0, 28.0, 65.0, 22.0, 60.0, 1.65, 2, 2),
('2024-03-20', 3, 60.0, 40.0, 85.0, 12.0, 85.0, 1.80, 1, 3),
('2024-01-05', 4, 58.0, 38.0, 82.0, 18.0, 78.0, 1.78, 3, 4),
('2024-04-12', 5, 52.0, 30.0, 70.0, 25.0, 65.0, 1.60, 2, 5),
('2024-05-20', 1, 56.0, 36.0, 78.0, 14.5, 74.0, 1.75, 1, 1),
('2024-02-10', 6, 62.0, 42.0, 90.0, 20.0, 90.0, 1.85, 4, 1),
('2024-03-15', 7, 54.0, 29.0, 68.0, 21.0, 62.0, 1.70, 1, 2),
('2024-04-22', 8, 65.0, 45.0, 95.0, 18.0, 95.0, 1.90, 5, 3),
('2024-05-05', 9, 57.0, 37.0, 80.0, 16.0, 76.0, 1.77, 3, 4),
('2024-06-10', 10, 53.0, 31.0, 72.0, 24.0, 68.0, 1.68, 2, 5),
('2024-01-20', 11, 59.0, 39.0, 83.0, 15.0, 80.0, 1.82, 1, 1),
('2024-02-25', 12, 51.0, 27.0, 64.0, 19.0, 58.0, 1.62, 4, 2),
('2024-03-30', 13, 63.0, 43.0, 88.0, 14.0, 88.0, 1.80, 5, 3),
('2024-04-05', 14, 60.0, 41.0, 86.0, 17.0, 82.0, 1.85, 2, 4),
('2024-05-15', 15, 49.0, 26.0, 62.0, 18.0, 55.0, 1.69, 1, 5),
('2024-06-20', 16, 64.0, 44.0, 89.0, 13.0, 84.0, 1.80, 3, 1),
('2024-01-15', 17, 56.0, 36.0, 81.0, 16.0, 77.0, 1.83, 4, 2),
('2024-02-20', 18, 61.0, 40.0, 87.0, 15.0, 86.0, 1.78, 5, 3),
('2024-03-25', 19, 58.0, 38.0, 84.0, 19.0, 79.0, 1.74, 1, 4),
```

```
('2024-04-30', 20, 52.0, 32.0, 71.0, 23.0, 63.0, 1.60, 2, 5),
('2024-05-10', 21, 66.0, 46.0, 92.0, 11.0, 92.0, 1.84, 3, 1),
('2024-06-15', 22, 68.0, 48.0, 94.0, 10.0, 98.0, 1.91, 4, 2),
('2024-01-25', 23, 59.0, 39.0, 85.0, 17.0, 81.0, 1.73, 5, 3),
('2024-02-10', 24, 62.0, 41.0, 88.0, 16.0, 83.0, 1.75, 1, 4),
('2024-03-12', 25, 50.0, 28.0, 66.0, 20.0, 59.0, 1.68, 2, 5),
('2024-04-18', 26, 57.0, 37.0, 82.0, 18.0, 75.0, 1.78, 3, 1),
('2024-05-22', 27, 61.0, 42.0, 86.0, 14.0, 84.0, 1.83, 4, 2),
('2024-06-25', 28, 55.0, 35.0, 79.0, 13.0, 72.0, 1.73, 5, 3),
('2024-01-30', 29, 63.0, 43.0, 89.0, 12.0, 89.0, 1.83, 1, 4),
('2024-02-28', 30, 54.0, 33.0, 74.0, 22.0, 64.0, 1.70, 2, 5);
```

3.4.5 CONSULTAS

Consulta 1 - Esta consulta gera uma visão geral do aluno, unificando dados administrativos (plano e pagamento), de acesso (status da catraca e Gympass) e de atendimento (recepcionista responsável).

```
SELECT
    aluno.nome_aluno,
    plano.nome_plano,
    maquinapagamento.status_pagamento,
    catraca.status AS status_catraca,
    recepcionista.nome_recepcionista,
    servidorgympass.status_validacao AS status_gympass
FROM Aluno aluno
    JOIN Plano plano ON aluno.cod_plano = plano.cod_plano
    JOIN Maquina_de_Pagamento maquinapagamento ON aluno.cod_m =
maquinapagamento.cod_m
    JOIN Catraca catraca ON aluno.cod_c = catraca.cod_c
    JOIN Recepcionista recepcionista ON aluno.cod_r =
recepcionista.cod_r
JOIN Servidor_gympass servidorgymppass ON catraca.cod_gympass =
servidorgympass.cod_gympass;
```

Consulta 2 - Mostrar os alunos que entram pela catraca, vinculados a recepcionistas, e validar o status do Gympass e o nome do plano que o aluno possui.

```
SELECT
    aluno.nome_aluno,
    recepcionista.nome_recepcionista,
    catraca.status AS status_catraca,
    servidorgympass.status_validacao,
```

```
    plano.nome_plano
FROM Aluno aluno
    JOIN Recepcionista recepcionista ON aluno.cod_r =
recepcionista.cod_r
    JOIN Catraca catraca ON aluno.cod_c = catraca.cod_c
    LEFT JOIN Servidor_Gympass servidorgympass ON catraca.cod_gympass =
servidorgympass.cod_gympass
    JOIN Plano plano ON aluno.cod_plano = plano.cod_plano;
```

Consulta 3 - Relatório de alunos externos (Gympass), mostrando se o acesso foi validado e
qual catraca utilizaram, com ligação ao plano e máquina de pagamento.

```
SELECT
    alunoexterno.cod_ae,
    servidorgympass.status_validacao,
    catraca.status AS status_catraca
FROM Aluno_externo alunoexterno
    JOIN Servidor_Gympass servidorgympass ON alunoexterno.cod_gympass =
servidorgympass.cod_gympass
    JOIN Catraca catraca ON servidorgympass.cod_gympass =
catraca.cod_gympass;
```

Consulta 4 - Obter todos os alunos, seus professores, a última avaliação feita, status do pagamento e recepcionista responsável.

```
SELECT
    aluno.nome_aluno,
    professor.nome_professor,
    MAX(avaliacao.data) AS ultima_avaliacao,
    recepcionista.nome_recepcionista,
    maquinapagamento.status_pagamento
FROM Aluno aluno
    JOIN Avaliacao_Fisica avaliacao ON aluno.cod_a = avaliacao.cod_a
    JOIN Professor professor ON avaliacao.cod_p= professor.cod_p
    JOIN Recepcionista recepcionista ON avaliacao.cod_r =
recepcionista.cod_r
    JOIN Maquina_de_Pagamento maquinapagamento ON aluno.cod_m =
maquinapagamento.cod_m
GROUP BY aluno.nome_aluno, professor.nome_professor,
recepcionista.nome_recepcionista, maquinapagamento.status_pagamento;
```

Consulta 5 - Listar todos os planos ativos, mostrando os alunos matriculados, a catraca que usam, se têm Gympass, o status do pagamento e o professor das avaliações.

```
SELECT
    plano.nome_plano,
    aluno.nome_aluno,
    catraca.status AS status_catraca,
    gympass.status_validacao,
    maquinapagamento.status_pagamento,
    professor.nome_professor
FROM Plano plano
    JOIN Aluno aluno ON plano.cod_plano = aluno.cod_plano
    JOIN Catraca catraca ON aluno.cod_c = catraca.cod_c
    LEFT JOIN Servidor_Gympass gympass ON catraca.cod_gympass =
gympass.cod_gympass
    JOIN Maquina_de_Pagamento maquinapagamento ON plano.cod_plano =
maquinapagamento.cod_plano
    LEFT JOIN Avaliacao_Fisica avaliacao ON aluno.cod_a =
avaliacao.cod_a
    LEFT JOIN Professor professor ON avaliacao.cod_p = professor.cod_p;
```

3.4.6 VISÕES – VIEWS

View 1 - Uma visão estatística que quantifica o volume de trabalho de cada recepcionista. Ela conta quantos alunos, professores, planos e máquinas de pagamento estão associados a cada funcionário da recepção, servindo como indicador de desempenho individual.

```
CREATE VIEW vw_resumo_recepcionistas AS
SELECT
    recepcionista.nome_recepcionista,
    COUNT(DISTINCT aluno.cod_a) AS qtd_alunos,
    COUNT(DISTINCT professor.cod_p) AS qtd_professores,
    COUNT(DISTINCT plano.cod_plano) AS qtd_planos,
    COUNT(DISTINCT maquinapagamento.cod_m) AS qtd_maquinas
FROM Recepcionista recepcionista
    LEFT JOIN Aluno aluno ON recepcionista.cod_r = aluno.cod_r
    LEFT JOIN Avaliacao_Fisica avaliacao ON aluno.cod_a =
avaliacao.cod_a
    LEFT JOIN Professor professor ON avaliacao.cod_p = professor.cod_p
    LEFT JOIN Plano plano ON aluno.cod_plano = plano.cod_plano
```

```
    LEFT JOIN Maquina_de_Pagamento maquinapagamento ON plano.cod_plano
= maquinapagamento.cod_plano
GROUP BY recepcionista.nome_recepcionista;
```

View 2 - Essa view funciona como um hub de verificação, centralizando todos os acessos via Gympass (tanto de alunos matriculados que vinculam a conta, quanto de externos). Ela permite à gerência monitorar falhas de validação ou acessos negados em tempo real.

```
CREATE VIEW vw_status_acesso_gympass AS
SELECT
    aluno.nome_aluno,
    alunoexterno.cod_ae AS aluno_externo,
    gympass.status_validacao,
    catraca.status AS status_catraca,
    plano.nome_plano,
    maquinapagamento.status_pagamento
FROM Servidor_Gympass gympass
    LEFT JOIN Aluno_externo alunoexterno ON gympass.cod_gympass =
alunoexterno.cod_gympass
    LEFT JOIN Catraca catraca ON gympass.cod_gympass =
catraca.cod_gympass
    LEFT JOIN Aluno aluno ON catraca.cod_c = aluno.cod_c
    LEFT JOIN Plano plano ON aluno.cod_plano = plano.cod_plano
    LEFT JOIN Maquina_de_Pagamento maquinapagamento ON plano.cod_plano
= maquinapagamento.cod_plano;
```

View 3 - Mostra o histórico de avaliações físicas dos alunos, com seus planos, pagamentos, recepcionistas e professores. Consolida o histórico de saúde do aluno (peso, altura) com sua situação administrativa (pagamento e plano).

```
CREATE VIEW vw_historico_avaliacoes AS
SELECT
    aluno.nome_aluno,
    avaliacao.data AS data_avaliacao,
    avaliacao.peso,
    avaliacao.altura,
    professor.nome_professor,
    recepcionista.nome_recepcionista,
    plano.nome_plano,
    plano.situacao AS situacao_plano,
    maquinapagamento.status_pagamento
FROM Avaliacao_Fisica avaliacao
```

```
    JOIN Aluno aluno ON avaliacao.cod_a = aluno.cod_a
    JOIN Professor professor ON avaliacao.cod_p = professor.cod_p
    JOIN Recepcionista recepcionista ON avaliacao.cod_r =
recepcionista.cod_r
    JOIN Plano plano ON aluno.cod_plano = plano.cod_plano
    JOIN Maquina_de_Pagamento maquinapagamento ON plano.cod_plano =
maquinapagamento.cod_plano;
```

3.4.7 - Triggers

Trigger 1 - Quando a máquina registrar pagamento como "Aprovado", o plano do aluno deve ficar “Ativo/Em dia” automaticamente.

```
CREATE FUNCTION trg_atualiza_plano_pagamento()
RETURNS TRIGGER AS $$
BEGIN
      IF NEW.status_pagamento = 'Aprovado' THEN
        UPDATE Plano
        SET situacao = 'Ativo'
        WHERE cod_plano = NEW.cod_plano;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_pagamento_aprovado
AFTER UPDATE ON Maquina_de_Pagamento
FOR EACH ROW
EXECUTE FUNCTION trg_atualiza_plano_pagamento();
```

Trigger 2 - Se o servidor Gympass retornar status “Invalido”, a catraca ligada a ele deve ser bloqueada automaticamente.

```
CREATE FUNCTION trg_bloqueia_catraca_gympass()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.status_validacao = 'Invalido' THEN
        UPDATE Catraca
        SET status = 'Bloqueado'
        WHERE cod_gympass = NEW.cod_gympass;
    END IF;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_gympass_invalido
AFTER UPDATE ON Servidor_Gympass
FOR EACH ROW
EXECUTE FUNCTION trg_bloqueia_catraca_gympass();
```

Trigger 3 - Impede que o aluno tenha mais de um plano ativo por vez.

```
CREATE FUNCTION trg_limite_plano_por_aluno()
RETURNS TRIGGER AS $$
DECLARE
    quantidade integer;
BEGIN
    UPDATE Aluno
    SET ultima_avaliacao = NEW.data
        WHERE cod_a = NEW.cod_a;
    RETURN NEW;

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_ultima_avaliacao
AFTER INSERT ON Avaliacao_Fisica
FOR EACH ROW
EXECUTE FUNCTION trg_atualiza_ultima_avaliacao();
```

Trigger 4 - Impede que uma avaliação física seja cadastrada no futuro.

```
CREATE FUNCTION trg_proibir_avaliacao_futura()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.data > CURRENT_DATE THEN
        RAISE EXCEPTION 'Erro: A data da avaliação não pode ser no
futuro.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER tg_data_avaliacao_futura
BEFORE INSERT OR UPDATE ON Avaliacao_Fisica
FOR EACH ROW
EXECUTE FUNCTION trg_proibir_avaliacao_futura();
```

Trigger 5 - Libera entrada pela catraca somente se o plano estiver ativo.

```
CREATE FUNCTION trg_verifica_plano_catraca()
RETURNS TRIGGER AS $$
DECLARE
    status_plano VARCHAR(20);
BEGIN
    SELECT situacao INTO status_plano
    FROM Plano
    WHERE cod_plano = NEW.cod_plano;

    IF status_plano <> 'Ativo' THEN
        UPDATE Catraca
        SET status = 'Bloqueado'
        WHERE cod_c = NEW.cod_c;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_plano_catraca
AFTER INSERT OR UPDATE ON Aluno
FOR EACH ROW
EXECUTE FUNCTION trg_verifica_plano_catraca();
```

3.4.8 - Storage Procedures

Storage Procedure 1 - Cadastra novo aluno completo.

```
CREATE PROCEDURE sp_cadastrar_aluno(
    p_cod_a INT,
    p_nome VARCHAR,
    p_cpf CHAR(11),
    p_data_nasc DATE,
    p_digital VARCHAR,
    p_cod_r INT,
    p_cod_c INT,
```

```
    p_cod_m INT,
    p_cod_plano INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO Aluno(cod_a, nome_aluno, cpf_aluno,
data_nascimento_aluno, digital, cod_r, cod_c, cod_m, cod_plano)
    VALUES(p_cod_a, p_nome, p_cpf, p_data_nasc, p_digital, p_cod_r,
p_cod_c, p_cod_m, p_cod_plano);
END;
$$;
```

Storage Procedure 2 - Regista o pagamento e atualiza o plano automaticamente.

```
CREATE PROCEDURE sp_registrar_pagamento(
    p_cod_m INT,
    p_status VARCHAR(20)
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE Maquina_de_Pagamento
    SET status_pagamento = p_status
    WHERE cod_m = p_cod_m;

    -- Trigger cuidará da ativação do plano
END;
$$;
```

Storage Procedure 3 - Consulta o status no servidor do GymPass e retorna se o aluno pode entrar.

```
CREATE FUNCTION fn_validar_gympass(p_cod_gympass INT)
RETURNS VARCHAR
LANGUAGE plpgsql
AS $$
DECLARE
    v_status VARCHAR(20);
BEGIN
    SELECT status_validacao INTO v_status
    FROM Servidor_Gympass
    WHERE cod_gympass = p_cod_gympass;
```

```
    IF v_status = 'Valido' THEN
        RETURN 'Acesso Liberado';
    ELSIF v_status = 'Invalido' THEN
        RETURN 'Acesso Negado';
    ELSE
        RETURN 'Erro na validação. Tente novamente.';
    END IF;
END;
$$;
```

Storage Procedure 4 - Registra a avaliação física de um aluno.

```
CREATE PROCEDURE sp_registrar_avaliacao(
    p_data DATE,
    p_cod_a INT,
    p_med_perna DECIMAL,
    p_med_braco DECIMAL,
    p_med_cintura DECIMAL,
    p_gordura DECIMAL,
    p_peso DECIMAL,
    p_altura DECIMAL,
    p_cod_r INT,
    p_cod_p INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO Avaliação Física(data, cod_a, med_perna, med_braco,
med_cintura, percentual_gordura, peso, altura, cod_r, cod_p)
    VALUES(p_data, p_cod_a, p_med_perna, p_med_braco,
p_med_cintura,p_gordura, p_peso, p_altura, p_cod_r, p_cod_p);
END;
$$;
```

Storage Procedure 5 - Permite à recepcionista atualizar os dados do aluno.

```
CREATE PROCEDURE sp_atualizar_aluno(
    p_cod_a INT,
    p_nome VARCHAR,
    p_cpf CHAR(11),
    p_data_nasc DATE,
    p_digital VARCHAR,
```

```
    p_cod_r INT,
    p_cod_c INT,
    p_cod_m INT,
    p_cod_plano INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE Aluno
    SET nome_aluno = p_nome,
        cpf_aluno = p_cpf,
        data_nascimento_aluno = p_data_nasc,
        digital = p_digital,
        cod_r = p_cod_r,
        cod_c = p_cod_c,
        cod_m = p_cod_m,
        cod_plano = p_cod_plano
    WHERE cod_a = p_cod_a;
END;
$$;
```

3.5 Dicionário de Dados - Projeto Academia

O dicionário de dados apresenta de forma detalhada as tabelas criadas, assim como seus atributos e restrições de integridade. As descrições das 9 tabelas propostas estão a seguir:

Tabela 1: Recepcionista

Descrição: Armazena dados dos recepcionistas.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_r	INT	Sim	Sim	Não	-	Código do recepcionista
nome_recepcionista	VARCHAR(100)	Sim	Não	Não	-	Nome completo
cpf_recepcionista	CHAR(11)	Sim	Não	Não	-	CPF único
salario_recepcionista	DECIMAL(10,2)	Não	Não	Não	-	Salário
data_nascimento_recepcionista	DATE	Não	Não	Não	-	Data de nascimento

Fonte: Os autores

Tabela 2: Recepcionista - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_rec_salario	CHECK	salario_recepcionista >= 0	Salário não pode ser negativo
ck_rec_nascimento	CHECK	data_nascimento_recepcionista > '1900-01-01'	Data lógica

Fonte: Os autores

Tabela 3: Servidor_Gympass
Descrição: Servidor de validação do Gympass.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_gympass	INT	Sim	Sim	Não	-	Código do registro Gympass
status_validacao	VARCHAR(20)	Sim	Não	Não	-	Status da validação

Fonte: Os autores

Tabela 4: Servidor_Gympass - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_gym_status	CHECK	status_validacao IN ('Valido','Invalido','Erro')	Status permitido

Fonte: Os autores

Tabela 5: Catraca
Descrição: Controla o acesso via catraca.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_c	INT	Sim	Sim	Não	-	Código da catraca
status	VARCHAR(20)	Sim	Não	Não	-	Estado da catraca
cod_gympass	INT	Não	Não	Sim	-	Referência ao Gympass

Fonte: Os autores

Tabela 6: Catraca - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_catraca_status	CHECK	status IN ('Liberado','Bloqueado','Manutencao')	Status permitido

Fonte: Os autores

Tabela 7 : Plano

Descrição: Planos de assinatura disponíveis.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_plano	INT	Sim	Sim	Não	-	Código do plano
nome_plano	VARCHAR(50)	Sim	Não	Não	-	Nome do plano
preco_plano	DECIMAL(10,2)	Sim	Não	Não	-	Preço
duracao	INT	Sim	Não	Não	-	Duração em dias/meses
situacao	VARCHAR(20)	Sim	Não	Não	-	Status do plano

Fonte: Os autores

Tabela 8 : Plano - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_plano_preco	CHECK	preco_plano >= 0	Preço não negativo
ck_plano_duracao	CHECK	duracao > 0	Duração positiva
ck_plano_situacao	CHECK	situacao IN ('Ativo','Inativo','Suspenso')	Status permitido
uq_plano_nome	UNIQUE	nome_plano	Nome único

Fonte: Os autores

Tabela 9: Maquina_de_Pagamento

Descrição: Registra status de pagamento.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_m	INT	Sim	Sim	Não	-	Código da máquina
status_pagamento	VARCHAR(20)	Sim	Não	Não	-	Status
cod_plano	INT	Sim	Não	Sim	-	Plano associado

Fonte: Os autores

Tabela 10: Maquina_de_Pagamento - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_maq_status	CHECK	status_pagamento IN ('Aprovado','Recusado','Pendente')	Status permitido

Fonte: Os autores

Tabela 11: Aluno
Descrição: Cadastro de alunos.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_a	INT	Sim	Sim	Não	-	Código do aluno
nome_aluno	VARCHAR(100)	Sim	Não	Não	-	Nome
cpf_aluno	CHAR(11)	Sim	Não	Não	-	CPF único
data_nascimento_aluno	DATE	Sim	Não	Não	-	Nascimento
digital	VARCHAR(200)	Não	Não	Não	-	Digital biométrica
cod_r	INT	Sim	Não	Sim	-	Recepcionista responsável
cod_c	INT	Sim	Não	Sim	-	Catraca usada
cod_m	INT	Sim	Não	Sim	-	Máquina de pagamento
cod_plano	INT	Sim	Não	Sim	-	Plano

Fonte: Os autores

Tabela 12: Aluno - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_aluno_nasc	CHECK	data_nascimento_aluno > '1900-01-01'	Data lógica
ck_aluno_digital	CHECK	LENGTH(digital) > 0	Digital não vazia

Fonte: Os autores

Tabela 13: Aluno_externo
Descrição: Alunos que utilizam Gympass.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_ae	INT	Sim	Sim	Não	-	Código
cod_gympass	INT	Sim	Não	Sim	-	Gympass

Fonte: Os autores

Tabela 14: Professor
Descrição: Professores cadastrados.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
cod_p	INT	Sim	Sim	Não	-	Código
cref	VARCHAR(20)	Sim	Não	Não	-	Registro profissional
data_nascimento_professor	DATE	Sim	Não	Não	-	Nascimento
salario_professor	DECIMAL(10,2)	Não	Não	Não	-	Salário
cpf_professor	CHAR(11)	Sim	Não	Não	-	CPF único
nome_professor	VARCHAR(100)	Sim	Não	Não	-	Nome

Fonte: Os autores

Tabela 15: Professor - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_prof_nascimento	CHECK	data_nascimento_professor > '1900-01-01'	Data lógica
ck_prof_salario	CHECK	salario_professor >= 0	Salário não negativo
ck_prof_cref_len	CHECK	LENGTH(cref) >= 4	Tamanho mínimo do CREF
ck_prof_salario_teto	CHECK	salario_professor < 50000	Teto salarial

Fonte: Os autores

Tabela: Avaliacao_Fisica

Descrição: Avaliações físicas de alunos.

Nome	Tipo	Não Nulo?	PK?	FK?	Default	Descrição
data	DATE	Sim	Sim*	Não	-	Data da avaliação
cod_a	INT	Sim	Sim*	Sim	-	Aluno
med_perna	DECIMAL(5,2)	Não	Não	Não	-	Medida da perna
med_braco	DECIMAL(5,2)	Não	Não	Não	-	Medida do braço
med_cintura	DECIMAL(5,2)	Não	Não	Não	-	Medida da cintura
percentual_gordura	DECIMAL(5,2)	Não	Não	Não	-	Gordura corporal
peso	DECIMAL(5,2)	Não	Não	No	-	Peso
altura	DECIMAL(5,2)	Não	Não	Não	-	Altura
cod_r	INT	Sim	Não	Sim	-	Recepcionista
cod_p	INT	Sim	Não	Sim	-	Professor

Tabela: Avaliacao_Fisica - RESTRIÇÕES

Nome	Tipo	Definição	Descrição
ck_av_gordura	CHECK	percentual_gordura BETWEEN 0 AND 100	Gordura válida
ck_av_peso	CHECK	peso > 0	Peso > 0
ck_av_altura	CHECK	altura > 0 AND altura < 3	Altura lógica
ck_av_perna	CHECK	med_perna > 0	Perna positiva
ck_av_braco	CHECK	med_braco > 0	Braço positivo
ck_av_cintura	CHECK	med_cintura > 0	Cintura positiva
ck_av_cintura_vs_braco	CHECK	med_cintura > med_braco	Coerência corporal
ck_av_data_passado	CHECK	data <= CURRENT_DATE	Data não futura

4. Conclusões

O desenvolvimento deste projeto de banco de dados para o sistema de gerenciamento de academias permitiu a aplicação prática dos conceitos fundamentais abordados na disciplina. Partindo da modelagem conceitual (Diagrama Entidade-Relacionamento) até a implementação física em SQL, foi possível compreender a importância de uma estrutura de dados sólida para a integridade de um sistema de informação.

Através da aplicação de mais de várias restrições de integridade via ALTER TABLE, o banco de dados foi configurado para atuar também como uma barreira de segurança, validando desde a maioridade de funcionários até a coerência de medidas antropométricas.

fsfsf Dessa forma, o sistema projetado não serve apenas como um repositório de informações, mas como um componente ativo na validação de processos, garantindo que a academia opere com dados confiáveis e consistentes para relatórios financeiros e acompanhamento da evolução dos alunos por meio das consultas e views criadas.

Referências Bibliográficas

Chen, Zhengxin. *Data Warehousing and Data Marts*. **Elsevier**, 28 Nov. 2004, pp. 521–533, <www.sciencedirect.com/science/article/abs/pii/B0122272404000368.>

Cochrane, R. (2000) “Practical Applications of Triggers and Constraints: Successes and Lingering Issues,” Very Large Data Bases.

“Database Integrity - an Overview | ScienceDirect Topics.” *Sciencedirect.com*, 2016, <www.sciencedirect.com/topics/computer-science/database-integrity> Accessed 21 Nov. 2025.

Doorn, Jorge Horacio, and Laura C Rivero. *Database Integrity: Challenges and Solutions*. IGI Global, 1 July 2001.

“Introdução Ao Conceito de Tablespaces | Oracle Brasil.” *Oracle.com*, 2019, <www.oracle.com/br/technical-resources/articles> Accessed 21 Nov. 2025.

Oracle. “What Is a Relational Database (RDBMS)?” *Oracle.com*, Oracle, 18 June 2021, <www.oracle.com> Accessed 21 Nov. 2025.

Pykes, Kurtis. “What Are ACID Transactions? A Complete Guide for Beginners.” *Datacamp.com*, DataCamp, 18 Feb. 2025, <www.datacamp.com/blog/acid-transactions.>

Song, Il-Yeol, et al. “A Comparative Analysis of Entity-Relationship Diagrams 1.” *Journal of Computer and Software Engineering*, vol. 3, no. 4, 1995, pp. 427–459, <www.cin.ufpe.br>

Trillo-Montero, D.; Cosano-Lucena, S.; Gonzalez-Redondo, M.; Luna-Rodriguez, J.J.; Santiago, I. Design and Development of a Relational Database Management System (RDBMS) with Open Source Tools for the Processing of Data Monitored in a Set of Photovoltaic (PV) Plants. Appl. Sci. 2023, 13, 1357. https://doi.org/10.3390/app13031357

Watt, Adrienne. “Chapter 8 the Entity Relationship Data Model.” *Opentextbc.ca*, BCcampus, 24 Oct. 2014, <opentextbc.ca/dbdesign01/chapter> Accessed 24 Aug. 2025.